PATENTS
112025-125
882

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re The Application of:
Frederick E. Niemi

Serial No. 09/346,789

Filed: July 2, 1999

For:
DYNAMIC CONFIGURATION AND UP-DATING OF INTEGRATED DISTRIBUTED APPLICATIONS

Examiner: Lewis A. Bullock, Jr.

Art Unit: 2126

Cesari and McKenna, LLP
88 Black Falcon Avenue
Boston, MA 02210
October 4, 2004

"Express Mail" Mailing-Label Number: EV 335 592 182 US

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

DECLARATION OF PRIOR INVENTION

TO OVERCOME CITED PUBLICATION UNDER 37 C.F.R. §1.131

PATENTS
112025-125
882

PURPOSE OF DECLARATION

1. This declaration is to establish completion of the invention of this application in the United States, at a date prior to March 20, 1998, that is the effective date of the prior art publication, U. S. Patent No. 6,185,611 which was filed on March 20, 1998, and which was applied by the Examiner.

2. The person making this declaration is the inventor.

FACTS AND DOCUMENTARY EVIDENCE

3. To establish the date of completion of the invention of this application, the following attached document is submitted as evidence:

"Running Java Classes Over There", authored by the undersigned affiant, no internal date in the document, however by looking at the properties of the document I can determine that it was created and modified in October 1997. As the document says, by that date I had working prototypes of the designs claimed in the above identified U. S. Patent Application.

4. From this document, written by the inventor, it can be seen that the invention in this application was made at least by the date of October 1997, which is a date earlier than the effective date of the reference.

DILIGENCE

5. Applicant acknowledges through this declaration that Applicant acted with diligence in the completion of the invention from the time of his conception, to a time just prior to the date of the reference, up to the filing of this application.

TIME OF PRESENTATION OF THE DECLARATION

This declaration is submitted prior to final rejection.

DECLARATION

6. I, Frederick E. Niemi, as the author of the attached document and the sole inventor of the present invention, hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

PATENTS
112025-125
882

SIGNATURE

7. Signature

Full name of sole inventor: Frederick E. Niemi

Inventor's signature: Frederick E. Niemi Date: 10-4-2004

Country of Citizenship: United States of America

Country of Residence: United States of America

Post Office Address: 101 Stanley Ct
Cary NC 27513

The following are my random thoughts on the list of topics we would like to discuss with the NMBU group.

09/346,789

Running Java Class Instances "Over There"

One technique I think we should exploit is running an instance of a Java class, application, applet on a workstation that is designated by the user or where required. This can be an instance of a particular type of Network Management Server application on a particular workstation, a class to be used to evaluate a event message or a diagnostic EUI to help determine a problem with a managed object.

Note: All three of these ideas has been prototyped.

Managed Object

ATTACH TO AFFIDAVIT OF FREDERICK NIEMI, INVENTOR

Everything we manage, routers, switch, hubs, bridges, network interface cards, DLSw Peers, IP stacks, etc. need to be represented by a Managed Object. A Managed Object is a logical representation of the things we manage. It provides the interfaces for other programs to deal with the object. It does the required polling to keep the Network management System aware of the current status of the thing being managed.

The interface provided include: (note: Tom George has ideas from past projects on additional commands)

get_diag_application - return an instance of an applet that the user interface can run to help diagnose this managed object.

activate - does whatever is necessary to activate this managed object

deactivate - does whatever is necessary to deactivate this managed object

diagnose - like demand poll in HP Openview/Netview for AIX. This runs some test to determine if there are any problems with this managed object. For example if a protocol running on top of IP determines there is a problem, it can issue this command to the IP Managed Object of the same network device to help determine what is wrong as quickly as possible.

etc.

get_hostname - host name of device

get_address - get address of device

get_community_name - read and write community names

etc.

Note: The idea of the managed object and the get_diag_application has been prototyped.

Event Distribution

These service needs to distribute event/messages from sources generating these event/messages to consumers that are interested in receiving these event/messages. The consumers should be able to specify the events/messages it is interested in receiving. It does this by creating an instance of a class that

implements a known interface. A method gets passed the event/message. The method evaluates the event/message to determine if the event/message is to be sent to the consumer. An instance of this class gets sent to the Event Distribution System where it runs and gets called for every event received. This class instance can be pushed all the way to the source and run in the sources Java process.

Note: Passing the class instance to the EDS process between the source and consumer has been prototyped. Pushing the class instance al the way to the source can be easily done and the EDS process does not have to have any event/messages running through it.

SNMP Poller

This service will be used by the managed object, data collector, threshold service, etc. to get or set SNMP data. This service should provide interfaces for get, get row, get table. There should be a way to tell what rate to get the data and use call backs to give the data back to the interested processes. The service should allow the changing of the polling rate, the table indexes and the MIB variables for the next poll.

Policy

Some ideas for what the Policy service can do:

> The user can specify that a particular threshold be applied to/for every interface of a given type. Then for every interface of this type discovered, the threshold is applied.
>
> If a particular threshold has been reached issue a particular command

A generic server interface needs to be defined that all servers need to support to allow policy rules to be applied as required.

Discovery

I like John Ahltrom's idea where based on the type of object, a Java class is instantiated. The name of the class is created based on the type of object. The discovery service also needs to define the topology.

Topology

The topology service needs to be based as some generic topology scheme. It should not contain any knowledge of how IP, VLANS, APPN etc. networks are hooked together. One scheme is based on some graphing theory. There are three different type of objects, graph, arc and vertex. A vertex is a single point, an arc connects graphs or vertex. And a graph contains graphs, vertex and arcs. From these notions, any topology and views can be depicted.

Note: A simple topology service containing vertex and arcs has been prototyped.

Data Collector

A schema can be defined through the interface. MIB variables as well as an analysis object used in thresholding can be used to determine what is to be stored. An API needs to be provided to allow queries of the data stored.

As in all uses of a database where the data continues to grow, a generic backup facility needs to be provided.

Threshold

Two thoughts:

1) An application can provide a list of MIB variables to get and an instance of a class to analyze the data and generate events when the data is out of spec.
2) An analysis object can be created to tell what data to retrieve and the formula to apply when all the data has been received. Then the results can be compared to a known value. The comparison can be things like: less than, equal to, has not changed, has changed, etc. A defined event can be generated. The same analysis can be used in several different comparisons and generated different levels of severity events.

Security

For each operator, we should create an object that defines the operators scope and span of control. For the scope of control, we should define the "role" of the operator. Every method call should define the role of the operator that can execute the command. We should look at modifying the stub files that are created when the IDL is compiled to check the scope or role of the operator issuing the command to determine if this operator can issue this method call. For span of control, we need to tie the domaining information we define for the servers to be the same format used in the operator information to determine what managed objects the operator can "see".

System Startup and Process Management

On every machine the user wants to run part of the Network management system, they need to install the server catcher program and the Visigenic osagent. This could be set up into some easy to install package. This server catcher should startup and run whenever the workstation is started. The user defines the domain for each server. This domain defines the bounds or parts of the network this Network Management Server is responsible for. An install utility is provided that allows the user to install each Network Management Server "package" on the designated servers. These packages consist of the server portions of the NM system like the discovery server or event distribution server etc. After the user selected where each of the servers are to run, the install server remembers that and makes sure upon startup that the servers are running on the designated server workstation. Each package should specify what other packages it is dependent on and the install server needs to make sure the servers are started in the correct order. The install server creates an instance of the server and runs it on the designated workstation. Then it can periodically poll the server to make sure it is running.

Note: Part of this has been prototyped.

Community Name Service

A community name service should be provided. One end allows the user to define what community names are defined for what IP address, IP address ranges and hostnames. Then when a new node is discovered, an application can get the defined community name for the agent. All this is just like the community name configuration for HP OpenView or NetView for AIX. What additions we should make is to read the topology service and all IP address in the same network device should resolve to the same community name. In HP OpenView or NetView for AIX, unless the user configures it properly, IP addresses from the

same network device can resolve to different community names; a mixture of defined as well as the default public.

Logging

I wonder if you can use a similar technique used by the data collector. A schema can be passed via an API and then the instance of the log can support storing of data as well as retrieval of the data through SQL expressions.

Persistence Store

All data is stored locally with the server.

Generic backup routines need to developed and used for any data that has the potnetial to grow without bounds. This includes logged data as well as collected data. This has the advantage to be faster to update and faster startup. Also, if there are network problems, you do not need to worry about losing connection to a centralized database.

There are two (2) database products I found that are written completely in Java. One is called JetStore by a company called XDB Systems and the other is called ObjectStore Persistence Storage Engine for Java by Object Design. Both are completely written in Java. We should look into using one of these products because of the ability of Java to write once and run anywhere. You do not have to worry about the right JDBC driver for the platform we are running on.

Reporting

Installation

See System Startup and Process Management.

Domain/Span of Control

See System Startup and Process Management.

Help

Through HTML pages. I wonder if you can point back to a page at cisco.com to get the most accurate up to date help pages.

User Interface

Java application/applet. It should be able to run in a browser as well as out of the browser.

Scheduler

Expand the scheduler used by the Poller to be able to do cron like jobs.

Relationships with NMP's

Replace them.

Scripting Language

Integration of Management Tools

Provide the generic MIB browser and graphers provided by HP OpenView or NetView for AIX.

Syslog Poller

Provide any data written to the Syslog to an interested application in real time. Applications register for the syslog data by providing a call back interface. When the data is available, the Syslog Poller calls the call back of the interested application with the data.

Message Catalog Support

Mark the embedded messages in the source. Run a pre-processor against the source to replace the marked messages with a reference to a catalog and create the message catalog.

Trap Receiver

receives a trap. Applications can provide a passed in class instance just like the event filtering in the EDS defined above. This class instance will generate an event from the trap received.

Diagnostic Schemes - Similar Debug

A global way to turn on and off debug for the system. A global debug logging facility to aid in debug.